Technical Forum

Gerry Wheeler
Computer Communications Network Group
University of Waterloo
Waterloo, Ontario CANADA

Undocumented M6800 Instructions

According to Motorola there are 197 valid operation codes for the M6800 microprocessor. This means that of the 256 possible 8 bit combinations, 59 are called invalid instructions.

Have you, like myself, ever wondered about these invalid codes? What would happen if you accidentally executed one? It does happen sometimes, of course, whenever your latest software creation takes an unexpected leap into never never land and begins executing randomly set memory locations. What are those holes in the op code chart anyway?

The mystery of those holes held my attention until the suspense was unbearable. To satisfy my gnawing curiosity I executed those codes deliberately, defying man and Motorola! And I got some interesting results.

Some of those codes seem to be just NOPS: they do nothing. Others change the flags in the condition code register according to some pattern that is, as yet, undeciphered.

But let me tell you about a couple of the interesting ones. See table 1 for descriptions of six instructions that Motorola didn't tell us about. The mnemonics are, of course, assigned by me.

The first one, NBA, is self-explanatory. The A and B accumulators are ANDed together, and the result is stored in A. I had to use NBA as the mnemonic because ABA is already used by Motorola. This instruction has been checked out thoroughly, and seems to be perfect, even setting the condition codes correctly. The only uncertainty is its execution time.

The store immediate instructions may require some explanation. Consider for a moment the load immediate instructions. These instructions take the byte following the op code and put it into the appropriate register. Therefore the store immediate instructions should store the register into the byte immediately after the op code, right? The only flaw is that there is a hole left after the instruction, and the register is stored after that (see figure 1). Note that the next instruction executed is the byte following the newly stored register. This means that the store immediate A and B instructions are three bytes long, and the store immediate X and SP instructions are four bytes long!

Now for the big surprise. This one has been dubbed HCF for Halt and Catch Fire. Well, almost. When this instruction is run the only way to see what it is doing is with an oscilloscope. From the user's point of view the machine halts and defies most attempts to get it restarted. Those persons with indicator lamps on the address bus will see that the processor begins to read all of memory, sequentially, very quickly. In effect, the address bus turns into a 16 bit counter. However, the processor takes no notice of what it is reading. . . it just reads. The only way out of this race is with the RESET line. The machine ignores the IRQ, NMI and HALT lines. For all intents and purposes the processor has halted and caught fire! It is quite possible that the HCF instructions are put into the 6800 design intentionally in the interest of production testing of newly fabricated processor chips.

Table 1: A list of six undocumented M6800 instructions and their definitions. The operations and operation codes which invoke them are defined in the column labelled Result, and the next instruction address is given in each case. Halt and Catch Fire (HCF) does not have a "next instruction" address because the processor hangs up.

Name	Mnemonic	Hexadecimal Op Code	Result	Next Instruction At
AND accumulators	NBA	14	A.B→A	PC + 1
store ACCA, immediate	STAA	87	A→PC+2	PC + 3
store ACCB, immediate	STAB	C7	B→PC+2	PC + 3
store SP, immediate	STS	8F	SPh→PC+2;SPl→PC+3	PC + 4
store IX, immediate	STX	CF	IXh→PC+2;IXl→PC+3	PC + 4
Halt and Catch Fire	HCF	9D or DD	see text	Not applicable

This one instruction might provide the automatic test equipment with a quick initial indication of whether the particular processor chip is a total dud, or a prospect for more detailed automatic testing and verification of defect free operation.

While these instructions are now documented, some warnings must in all fairness be stated lest the user run into problems. The primary warning is that there may be a reason that they were left undocumented: they may not work with every 6800 processor, so any software intended for production, distribution to friends or for publication should *never* use these instructions. At different times during the history of M6800 production at Motorola, revisions and changes in the production masks may alter the effects of these instructions without any warning to users; after all, an undocumented instruction is not there from Motorola's point of view, so why tell the users about changes in its definition? Similarly, when 6800 parts are acquired from suppliers other than Motorola, use of independent designs for the production masks by the second source leaves definition of these undocumented instructions unspecified and not necessarily identical to Motorola's definitions. But these warnings apply only to programs to be distributed in some way; if your personal processor executes these instructions and you find a use for them in your own handcrafted assembly language programs, then by all means take advantage of them.■

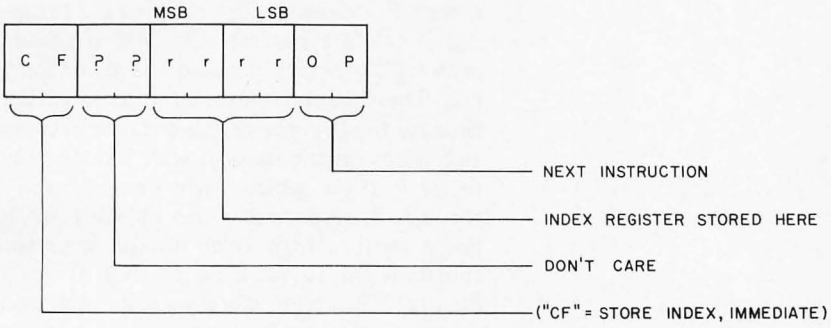

Figure 1: The "Store Index Immediate" instruction requires four bytes of memory, as illustrated here. The operation code hexadecimal CF is followed by one byte which is "don't care" as far as the operation of this instruction is concerned. The third and fourth bytes of the operation receive the 16 bit address value from the index register in the normal order. In this diagram, rrrr is the 16 bit target for the immediate store, and OP is the first byte of the next instruction. Operation of the "Store Stack Pointer Immediate" instruction is similar.